95-702 Distributed Systems

Cryptography and Security

Goals

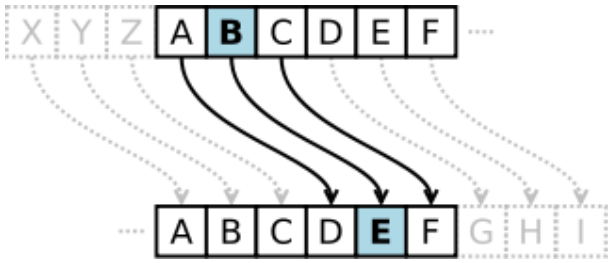
- Understand basic cryptography and security terms
- Understand security in terms of:
  - Secure web transmission
  - Authentication
    - Who are you?
  - Authorization
    - What are you allowed to do?
  - Certificates, and Digital Signatures
    - Is this document / software / transaction real?
- Have a basic understanding of the underlying theory and math behind web security.

2 kinds of cryptographic systems

- Symmetric key
  - Simple e.g. Caesar cipher
  - Most prevalent: AES
- Asymmetric key
  - Also known as Public Key (or Private / Public Key)
  - Most prevalent: RSA

Julius Caesar (shift) cipher

- Pick a key (a number)
- Shift the letters of the plaintext by the key to create the ciphertext.
- E.g.
  - Plaintext: Yellow cake
  - Key: 3
  - Ciphertext: Bhoorz fdnh



Source: http://en.wikipedia.org/wiki/File:Caesar3.svg

Caesar cipher

- Secret key algorithm
  - The sender and the receiver share a secret key
- Symmetric algorithm
  - Trivially-related keys are used to encode and decode the message
    - Trivially-related: uses the same key, or keys require only a simple transformation.
    - E.g. Caesar cipher: (English) symmetric key is 26-key

Brute Force Attack

- One way to discover the plaintext is to exhaustively try every possible key.
- Is the algorithm susceptible to being broken by exhaustively trying possible keys?
- Is the Caesar cipher susceptible to brute force attack?

Brute Force Attack Countermeasures

- Use extremely large keys
  - An 8 bit key has 2^8 possible keys
    - 256
  - A 16 bit key has 2^{16} possible keys
    - 65,536
  - A 32 bit key has 2^{32} possible keys
    - 4,294,967,296
  - A 256 bit key has 2^{256} possible keys
    - $1.15792089 \times 10^{77}$

How can we "mess up" the data better?

- Cryptography is essentially trying
  - to "mess up" the original data as much as possible
  - so that someone else cannot find the original
  - but be able to get the original data back with a key
- Encryption = "mess up"

- So how can we "mess up" the data better than the Caesar cipher does?
  - The blocks of plaintext are one character.
  - There is only so much you can do to one character.
  - So how about encrypting blocks of data?

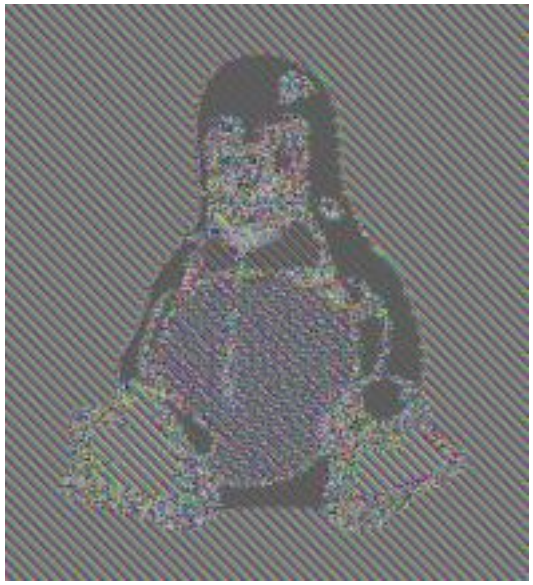
Block cipher

- Symmetric key cipher
- Works on *blocks* of text
  - E.g. 128 bit blocks
- Simple Caesar example: Two characters, Key: 1
  - cake (c=3, a=1, k=11, e=5)
  - 000110000101011100101
  - 000110001001011100110

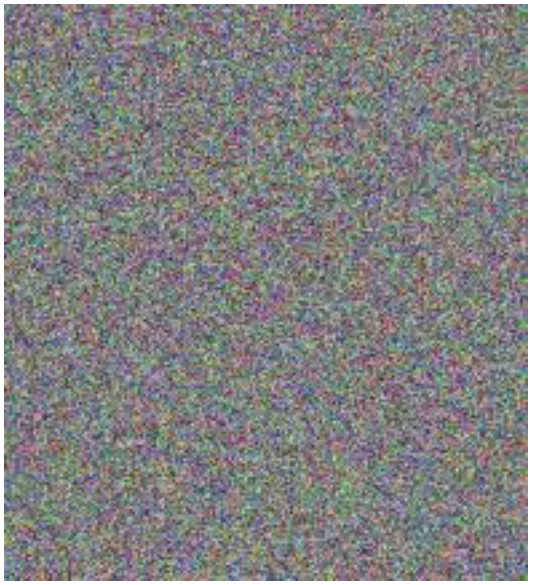
Shortcomings of block cipher

- If blocks' plaintext are identical, then their ciphertext will also be identical.
- This can be seen visually in the following 3 pictures.
- What you want it the 3rd picture.
- One way to do so is to mess up the current block with info from all prior blocks.

Original



Block Cipher



Block Cipher Chaining

Image source: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Block cipher chaining

- Each block is XOR'ed with the previous block before encrypting
  - The first block is XOR'ed with an initialization vector
  - The initialization is typically a random number
- Therefore each block is dependent on all plaintext blocks up to that point

01001 - initialization vector
00001 - key
4 bit block (e.g. one character)
cake
0001100001010110 0101 - encoded plaintext
01010 - xor c
01011 - encrypt c
01010 - xor a
01011 - encrypt a
00000 - xor k
00001 - encrypt k
00100 - xor e
00101 - encrypt e
0101101011000010 0101 - encrypted ciphertext

XOR				
0	^	0	=	0
0	^	1	=	1
1	^	0	=	1
1	^	1	=	0

Advanced Encryption Standard

- Very complex chaining block cipher
- Symmetric algorithm
- Adopted by US National Institute of Standards
  - Replaced the former standard: DES
    - Data Encryption Standard – 56 bit key block cipher
- Check what your browser uses:
  - In Chrome, browse to some https://… site
  - Click on padlock, and choose "Connection" tab

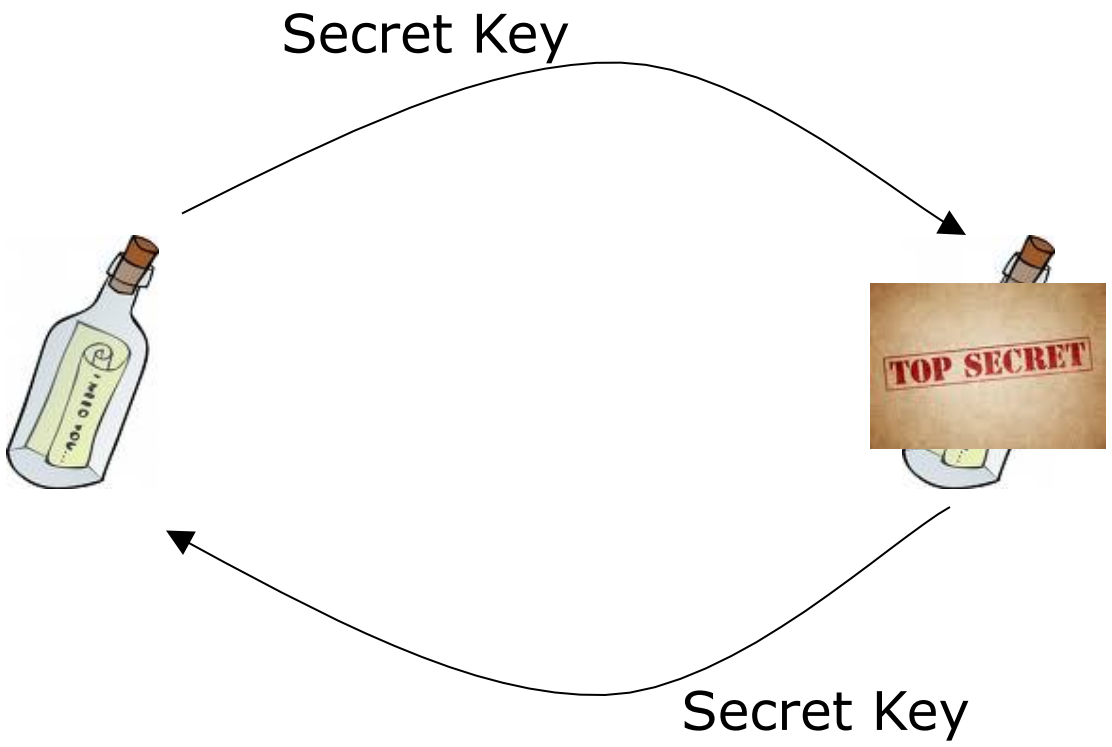
Recall Our Cast of Characters

- Alice and Bob
  - try to communicate over insecure channels.
- Eve
  - tries to view messages she should not be viewing.
- Mallory
  - tries to manipulate messages and be disruptive

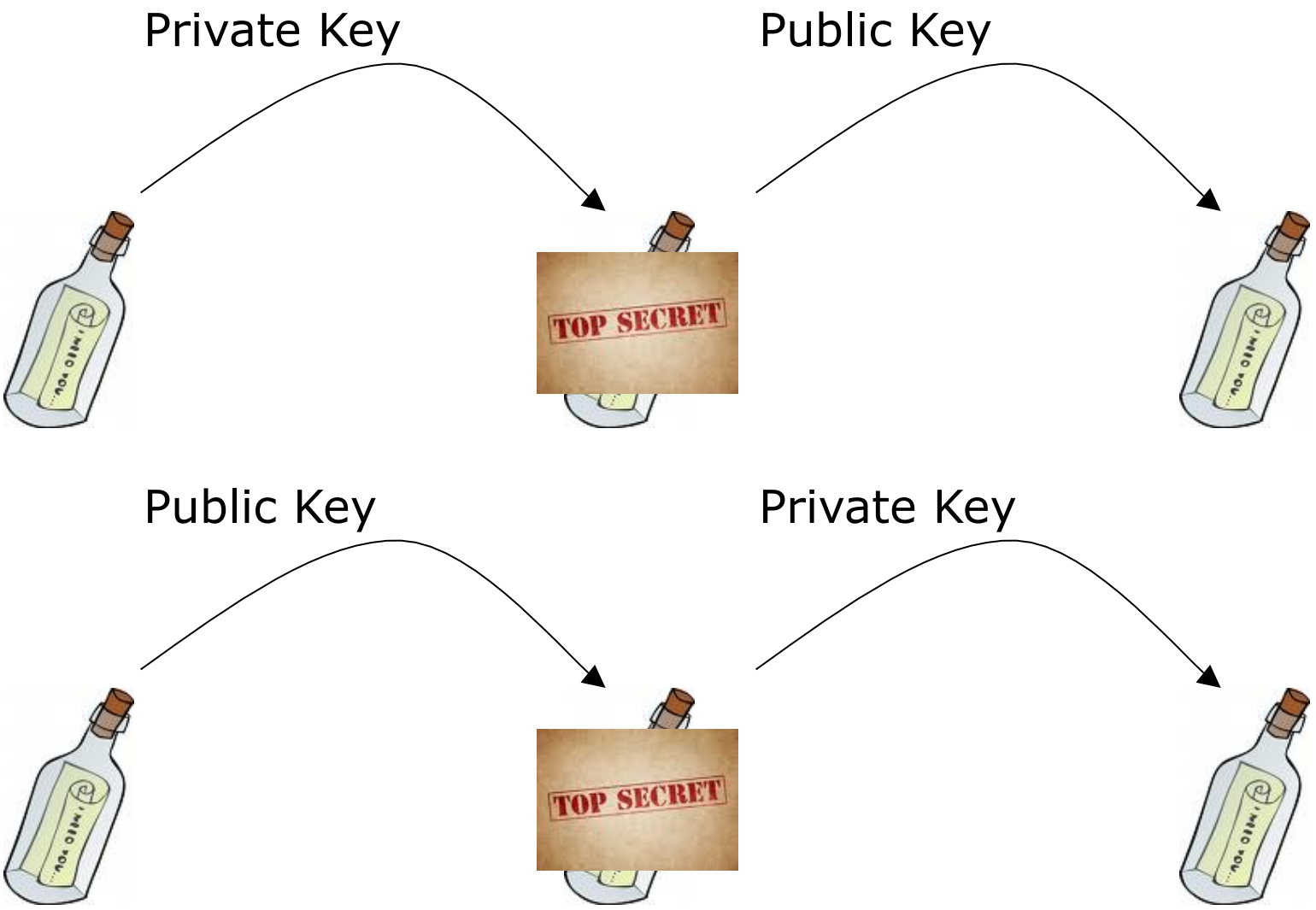
Symmetric vs Asymmetric algorithms

- Symmetric algorithms require Alice and Bob to share a secret (the key).
  - Both can encrypt and decrypt messages
- Asymmetric algorithms allow for not sharing a secret.
  - Alice can encode a message for Bob
  - She cannot decode messages already encoded for Bob

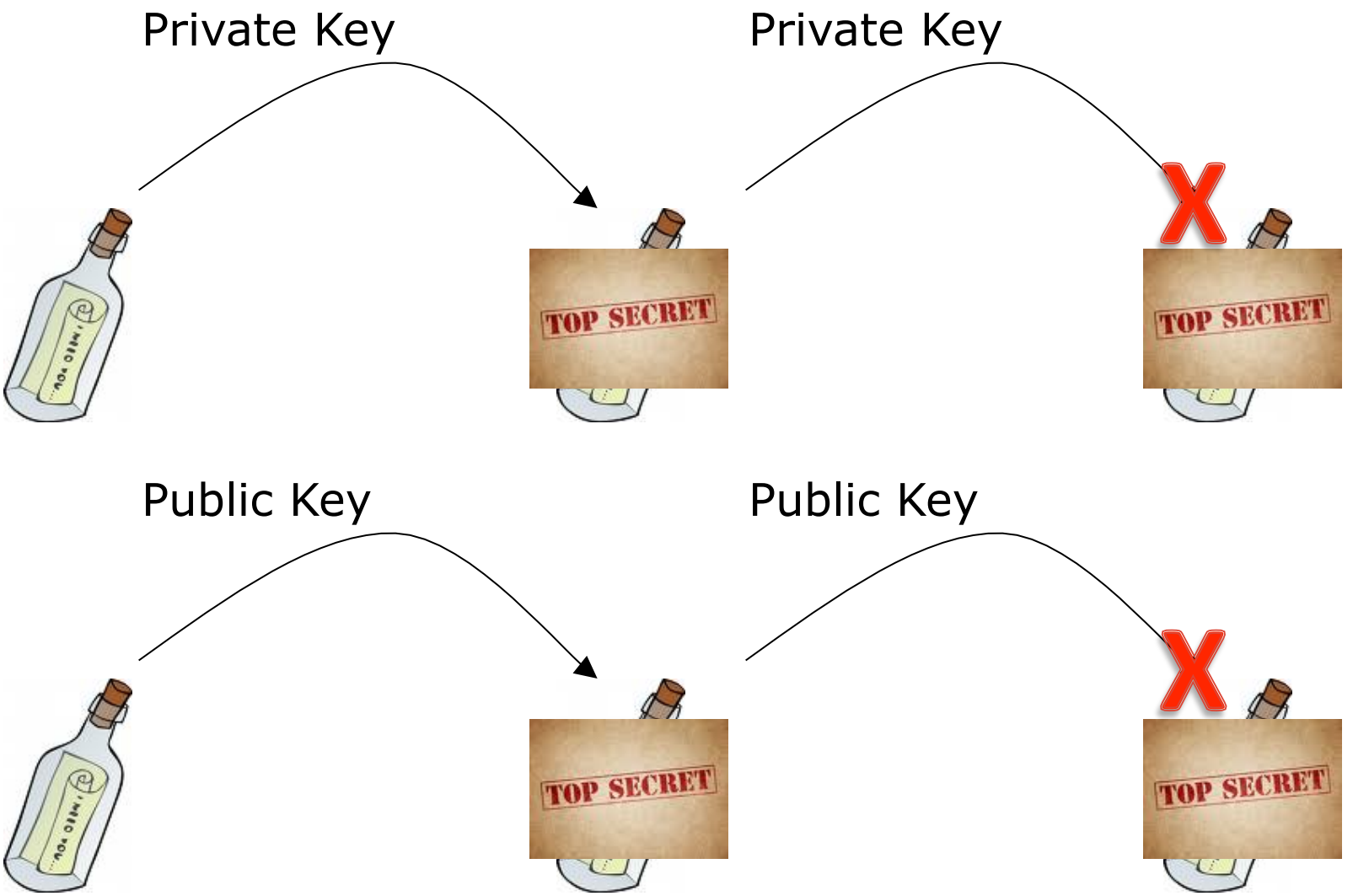
Symmetric



Asymmetric


Private Key
Public Key
TOP SECRET
Public Key
Private Key
TOP SECRET

Asymmetric


Private Key
Private Key
TOP SECRET
TOP SECRET
Public Key
Public Key
TOP SECRET
TOP SECRET

Public Key

- Uses asymmetric keys
- Single public key
  - Let the world see
- Single private key
  - You keep secret

RSA

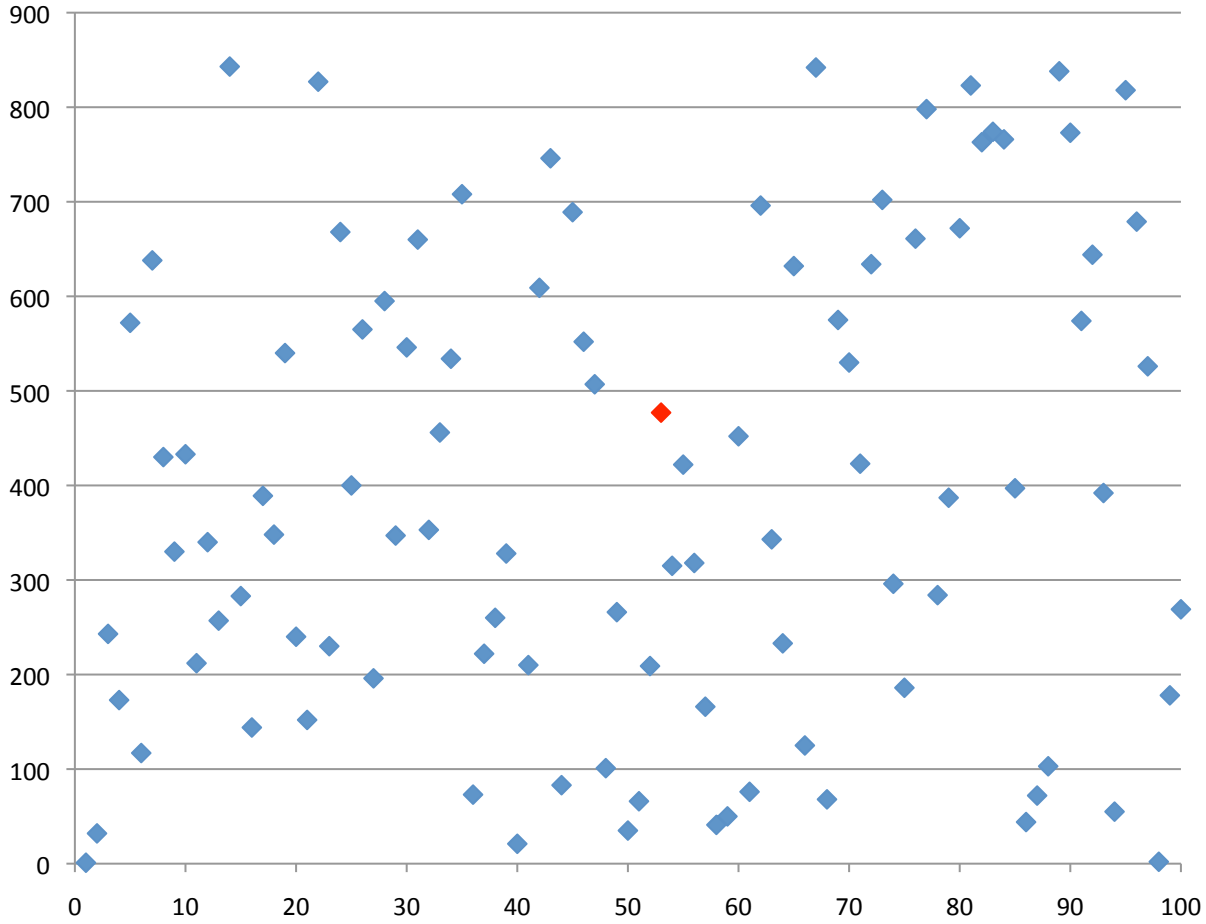
- RSA is a common public key encryption algorithm
  - Named for its authors: Rivest, Shamir, & Adleman
- Developed in 1977
- Based on the mathematics of large prime numbers
  - If you have the numbers, you can use them to encrypt messages
  - If you don't have the numbers, it is infeasible to guess them

The RSA algorithm

- There is a nice tutorial tool at:
  - https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- You can practice by using prime numbers you can find at:
  - http://en.wikipedia.org/wiki/List_of_prime_numbers
- E.g. 23, 37

Plot of 1-100 encrypted with RSA key {5,851}

- $y = x^5 \% 851$
- See a pattern?
- Can you predict the value at 101?
- Unpredictability is its strength
- Red dot is at (53, 477)
- Corresponding key is {317,851}
- $477^{317}\%851=53$



Two Roles of Private and Public Keys

1. Encryption / Decryption
   - Public key
     - Used by others
     - To encrypt a message intended only for you
   - Private key
     - Used by you
     - To decrypt a message originally encrypted by your public key

TLS/SSL uses public and private keys in this way

Your browser encrypts the AES key with Amazon's public key

Amazon decrypts the AES key with its private key

2. Signing / Verification
   - Private key
     - Used to sign a document so that others can verify the source
   - Public key
     - Used to verify that a signed document was signed by you.

Cryptographic protocols

- Cryptographic protocols build on the use of cryptographic algorithms
- Two prominent protocols:
  - Transport Layer Security (TLS) (newer)
  - Secure Socket Layer (SSL) (older)
- Both are application-level protocols, that work above the transport layer (especially TCP) to provide safe end-to-end communication.
- Often folk will refer to secure communication as "over SSL" regardless of whether TLS or SSL is being used.

Secure email & web

- SMTP and IMAP (email protocols) can work above TLS or SSL to provide secure email transmission
- HTTPS is HTTP over TLS/SSL to provide secure web communication
- There will be more on cryptographic protocols next class.

Symmetric vs Asymmetric algorithms

- Symmetric
  - Fast
  - Difficult to distribute and and keep keys secure
- Asymmetric
  - 100 to 1000 times slower
  - Can allow for public keys
- TLS / SSL uses the best of both worlds:
  - Use asymmetric keys to exchange symmetric keys at the beginning of a conversation
  - Symmetric keys will have the lifespan of that conversation

TLS / SSL Protocol Handshake

- Start with RSA Public / Private keys
  - Ask Amazon for their Public key
  - Amazon replies with their Public key
- Generate a 128 bit (or bigger) random number
  - This is your "session" new AES key
  - Encrypt the new AES key with the Amazon Public Key
  - Send the encrypted key to Amazon
    - Notice you are sending an AES key encrypted with a RSA key
- Amazon receives the encrypted key
  - They (and only they) can decrypt the AES key with their RSA Private key
  - They now have the AES key you created for this session
- Amazon and your browser communicate by encrypting and decrypting all messages with the same AES 128 bit random-number key.
  - At the end of this session, both forget the AES key

Two Roles of Private and Public Keys

1. Encryption / Decryption
   - Public key
     - Used by others
     - To encrypt a message intended only for you
   - Private key
     - Used by you
     - To decrypt a message originally encrypted by your public key

2. Signing / Verification
   - Private key
     - Used to sign a document so that others can verify the source
   - Public key
     - Used to verify that a signed document was signed by you.

E.g. Verifying authenticity

- How can you guarantee to someone that a document you sent them is from you, and has not been changed?
- How can you guarantee that the software you are using came from Microsoft, and that it has not been altered?
- You want to keep the document / software / image / etc. viewable and usable, but just want a scheme by which others can verify its authenticity.

Digital Signatures

- Public key encryption can be used to provide digital signatures to validate authenticity
- Digital signatures are better that real signatures, for people can alter a paper document once you have signed it.
- With digital signatures, if the document is changed, then the signature becomes invalid.
- This is because the signature is a number based on the content of the document.
  - Or more specifically, on the *hash value* of a document.

Hash Functions

- Remember you calculated hash values in Project 1!
- Takes a file (application, document, picture, etc.)
- Returns a large, but fixed-size number.
- Any intentional or accidental change in the file will change its resulting hash value.
- The file being encoded is called the “message”
- The resulting hash value is also called the
  - "Message Digest"
  - Or simply "Digest"

Properties of a Good Hash Function

- For any message, the hash value is easy to compute.
- It is infeasible to create a new message that has a given hash value
  - I.e. though you know the hash value, you can't create a document to match that hash value
- It is infeasible to modify a message in any way without changing its hash value
  - All modifications change the hash value.
- It is completely unlikely that two documents will have the same hash value
  - So you don’t have to worry that Mallory will just be lucky and find another document with the same hash value

Common Hash Functions

- SHA1
  - Designed by the National Security Administration (NSA)
  - 160 bit digest
- MD5
  - 128 bit digest

- Both have some weakness.
  - Often you will see both used.
- Both SHA1 and MD5 are often represented as strings of hex digits
  - (As you did in Project 1)

Is a hash function encryption?

- Can a hash function be used to encrypt a message?

How digital signatures work

- Take your document (email message, etc)
- Calculate a hash function on it.
  - e.g. SHA2
- Encrypt the resulting hash value with your private key.
- The encrypted hash value is the digital signature.
- Send it …

How digital signatures work

- … The recipient
- Receives the document (email message, etc) and the digital signature.
- Decrypts the signature with _____ resulting in _____
- Calculate a hash of the document & compare it with the sender's hash value
- Should they be equal? Why?

How digital signatures work

- What does it mean if the hash values are not equal?
- Could Mallory change the document?
- Can Mallory change the document without changing the hash value?
- Can Eve read the document (email, etc.)?
  - What can you do about that?

Can we trust we have the right public key?

- We saw we could use:
  - Asymmetric keys (RSA)
  - To share a symmetric key (AES)
  - Then pass messages back and forth efficiently using the symmetric key
- This is essentially SSL
- So, I can use this scheme to safely send Amazon.com a message with my credit card number, correct?

Can we trust we have the right public key?

- Do I really know who I've been negotiating with?
- Is it really Amazon.com?
  - Or Mallory?
- They sent me their public key to use,
  - So if I knew that this was really Amazon.com's public key,
  - Then I could trust I'm working with the real Amazon.com,
  - For only Amazon.com has the corresponding private key.
- How do I know if I really have Amazon.com's public key?

Digital Certificates

- A Digital Certificate is a document that provides information about an organization
  - Most importantly, its public key
- And the Digital Certificate is digitally signed by some trusted party.

Digital Certificates

- Issued by trusted entities
  - Company IT Department (internally)
  - VeriSign
  - Thawte
  - Lots of others
- Typically contains
  - Owner's name
  - Owner's public key
  - Expiration date
  - Name of certificate issuer
  - Serial number
  - Issuer's digital signature
- E.g. Blackboard Digital Certificate

Review Scenario

- A while back, I got an email message from 2co.com saying my credit card had expired.
  - 2co.com is apparently the billing company, not the web hosting company I contract with.
- I needed to give them updated information, or my hosting account would be cancelled.
- I wondered: Is this real? Is it a phishing scam?
- I checked my history of email messages stating my monthly bill had been paid, , and they were consistently from a 2co.com address.
  - I concluded that 2co.com is legitimately the company that manages billing for my hosting company.
  - So I would go to 2co.com, and give them my updated credit information.

2co.com

- I pasted the https://2co.com URL into my browser.
  - What could go wrong?
  - Is the web site I receive really 2co.com?
  - Could it be an imposter?
  - How can I be sure?
- As part of the https handshake, it gave me its public key
- If it really is 2co.com's public key, then that would allow secure communication
- How can I check if I really have 2co.com's public key?
  - And not an imposter pretending to be 2co.com

How to establish trust of 2co.com

- How can I check if it is 2co.com's public key?
- It would be nice to have someone I trust
- Someone that would know what 2co.com's public key is.
- They could validate that the public key I have, supposedly from from 2co.com, is valid.

Certificate Authority

- (or Certification Authority)
- Also know simply as: CA
- If I know the public key of a CA, then I can decrypt a digital certificate from 2co.com
- 2co.com gets a digital certificate issued by the CA
  - It contains 2co.com's public key
  - It is digitally signed with the CA's private key
- Therefore my browser can:
  - Decrypt the digital signature of the certificate with the CA's public key
  - Perform a hash function on the certificate
  - Compare the computed hash value, with the hash value in the decrypted digital signature.
  - If they are the same, then CA has validated that 2co.com's public key, contained in the digital certificate, is authentic. And I can trust that I am actually communicating with 2co.com (not some imposter).

How did 2co.com get a digital certificate?

- 2co.com pays the CA for the service
- 2co.com has to demonstrate to CA that it is who it says it is.
  - E.g. By having other trusted information, such as a credit card in 2co.com's name.
    - (The CA should *not* issue me a digital certificate authenticating a public key I generate as bnymellon.com!)

DigiNotar

Fraudulent Google credential found in the wild

Did counterfeit SSL cert target Iranians?

World ostracizes firm that issued bogus Google credential

DigiNotar says it was breached ... but little else

By D

Poste

Free

A co

resul

base

abou

the mear

pages or

Start

Withdraw

root certificates

BAPI

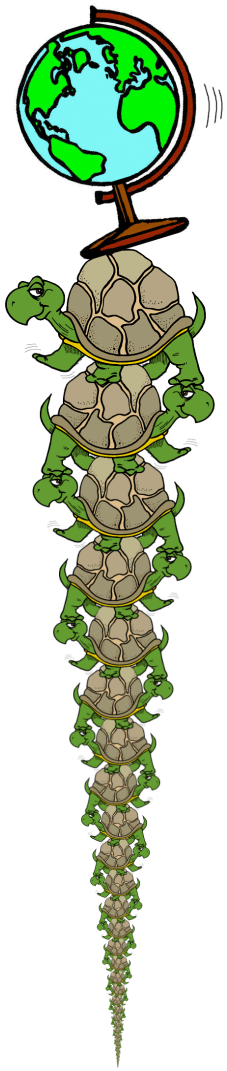
Bankruptcy report, 31 October 2011

Bankruptcy
Company DigiNotar BV
Number F 11 415
District Court of Haarlem
Date 09/20/2011

2co.com digital certificate

- Once 2co.com has demonstrated that it ***really is*** 2co.com, then the CA will take 2co.com's public key, put it in a certificate, and digitally sign it with the CA's public key.
- It is this certificate that 2co.com sends to browsers who contact it with https:..

But how can I trust that the CA is the CA?



- Is it Turtles all the way down?
- Installed in your laptop are CA **root certificates**
  - They came installed.
  - Or you deliberately installed them.
- You have a root certificate for common CAs
  - Thawte
  - Verisign
- Show on Mac, similar in Windows

So to work back up from the turtles

- I have a root certificate for Thawte on my laptop.
  - It contains Thawte's public key
- When I try to connect to 2co.com using https, 2co.com sends me a digital certificate they purchased from Thawte
- I can check the authenticity of 2co.com's digital certificate with Thawte's public key
- With 2co.com's public key, I can create a share secretly a new AES key.
- 2co.com and I can then communicate securely using this AES key.

Where are we now?

- Does 2co.com know who I am?
- Not yet. It knows it is communicating with someone securely, but not who it is communicating with.
- I know I am communicating securely with 2co.com, so it is safe to give my login and password.

Authenticate

- Now that we are communicating using SSL via https, I can securely send a login and password.
- Could someone jump into the middle at this point?
  - No, for we are using a symmetric key and AES
  - They would not know what the key is.
- 2co.com can check the login and password against their records, and decide if I really am who I say I am.

Review Questions

- The “padlock” on a browser means _____?
- Does the padlock means you are logged in?
- If you **are** logged into Amazon.com, which are true?
  - Your browser received Amazon’s public key
  - Your browser used a certificate authority to validate Amazon’s public key
  - Amazon validated your public key with a certificate authority
  - RSA was used between your browser and Amazon.
  - Your browser and the certificate authority are sharing a secret key
  - Your browser and Amazon are sharing a secret key.

Review Questions

- RSA is a {protocol or encryption algorithm}.
- AES is a {protocol or encryption algorithm}.
- SSL is a {protocol or encryption algorithm}.
- MD5 is an {encoding or encryption} algorithm.
- SHA1 is an {encoding or encryption} algorithm.

Custom Authentication

- You can create your own authentication
- Require a secure channel (e.g. SSL, TLS)
- Create a login page
  - User ID and Password sent to the server
  - Store in your database
    - userID as key
    - hashed password as value
  - Compare hashed submitted password against stored value
  - In this way, can't accidently disclose passwords

Authorization

- Authorization is different than Authentication
- Authentication establishes identity
- Authorization establishes access rights